UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/604,987	06/28/2000	Srivatsan Parthasarathy	MS146910.1	6447

27195 7590 04/21/2005

AMIN & TUROCY, LLP
24TH FLOOR, NATIONAL CITY CENTER
1900 EAST NINTH STREET
CLEVELAND, OH 44114

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

DATE MAILED: 04/21/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	**Application No.** 09/604,987	**Applicant(s)** PARTHASARATHY ET AL.
	Examiner Tuan A. Vu	**Art Unit** 2193

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *17 December 2004*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-27 and 29-35* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-27 and 29-35* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)	4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)	5) ☐ Notice of Informal Patent Application (PTO-152)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.	6) ☐ Other: _____.

[illegible]

DETAILED ACTION

1. This action is responsive to the Applicant's response filed 12/17/2004.

As indicated in Applicant's response, claims 1-3, 5-12, 14-18, 22-23, 27, 30 have been amended, and claim 28 canceled. Claims 1-27, 29-35 are pending in the office action.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-8, 18, and 23-25 are rejected under 35 U.S.C. 103(a) as being unpatentable over Renaud et al., USPN: 5,958,051 (hereinafter Renaud), in view of Shaw, USPN: 2002/0026634 (hereinafter Shaw).

As per claim 1, Renaud discloses a method for integrity checking employable by application programs at runtime (e.g. Fig. 6; col. 9, line 40 to col. 10, line 37), such method comprising:

providing an assembly (e.g. *data structure 300* – Fig. 3a) that contains a list of versioned modules (e.g. Fig. 3a-b; *data files 304-314, version of the file* - col. 6, lines 46-64; col. 7, lines 31-36 – Note: Java class files or applets are equivalent to modules) that make up the assembly;

providing a manifest (e.g. *signature file 302* – Fig. 3b) with a hash value of data related to at least one module of the list of modules (e.g. *identifier, one-way hash* - col. 7, lines 15-27);

comparing one module in the manifest at runtime to identify whether a runtime version of such module is similar the a version as provided from the source (e.g. *signature file ... author, version of file* – col. 6, line 53 to col. 7, line 5; steps *402-414* –Fig. 4).

But Renaud does not specify that the hash of the list of modules making up assembly is a hash of the contents of at least one module. Renaud, however, teaches arranging modules or file data in such way that digital signature is combined with digest encryption algorithm and that the manifest can include data related to the modules comprising the assembly (e.g. *data relating to each data file, MD5, MD2* - col. 7, lines 24-54) to facilitate processing and security checking. Shaw, in a system to receive application program or objects into an application client environment like Renaud's using a manifest of modules listed for activation of client-side applets(Renaud: Fig. 5a), discloses manifest of hashes representing code segments as checked from a list, each code segment being used and activated after being verified based on such hash checklist (e.g. pg. 3 - para 0034-0035). It would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the integrity checking of manifest-listed modules by Renaud such that such manifest lists not only the data modules hashed identifiers or data related to those modules but also a hash of the contents of such modules just as taught by Shaw, Renaud's hashing of module identifier differs from Shaw's module content hashing in that the hash by Shaw can store source provider identification. One ordinary skill in the art would be motivated to implement a manifest of hashed content or modules because via hashing and accompanying it with additional source identification, the modules not only can be verified for integrity but also can originate from more than one locations and then loaded separately (see

Shaw, para 0034) notably since the source identification or publisher site has been a security concern in Renaud's approach also (see col. 11-12 related to Fig. 5).

Nor does Renaud explicitly teach comparing hash of one module obtained to identify whether is similar to a version at build time of the assembly. Based on the author, the version information of the signature file as mentioned above; and the teaching by Renaud concerning the security level concerning site publisher using certificate (col. 11-12); and the hash content in a manifest enabling downloading from verifiable source locations as taught by Shaw, would have been obvious for one of ordinary skill in the art at the time the invention was made to implement the comparing of hash content, from a manifest list as by Renaud or Shaw, is based on whether it matches the source provider established identification, one such identification being a version of a build of such module as intended via the version teaching by Renaud (col. 6, line 53 to col. 7, line 5; steps 402-414 –Fig. 4). The motivation would have been obvious because of the level of trustiness raised concerning a version and authoring source as shown by Renaud; and secondly to enable correct update of versioned code as intended by Shaw's using hash content (e.g. para 0041, pg. 3)

As per claim 2, in view of the combined teachings of Renaud and Shaw from claim 1, the limitation of providing a hash of the content such that to provide manifest of hash of each modules that constitute the assembly would have been obvious because Shaw's integrity checking per module via hash content listed from a table as set forth above enables Renaud with the integrity checking for version of each module before update/use as well as trust-worthiness checking of source provider as mentioned in the rationale set forth in claim 1 above.

As per claim 3 and 4, Renaud further discloses providing identity information in the manifest (e.g. Fig. 3b, *additional data* – col. 6, lines 55-64) as well as publisher and version information.

As per claim 5, in reference to claim 1, Renaud discloses hash of the identifiers of versioned modules in the assembly and suggests placing a signature (or hashed representation)of the manifest itself on top of the manifest with all other module identifiers at the end of the manifest (e.g. Fig. 3B), but **does not disclose** providing a hash of the contents of such assembly at the end of the assembly. But this limitation to provide hash of the contents of modules has been addressed in claim 1 above. Hence it would have been obvious for one of ordinary skill in the art at the time the invention was made to add such hash of contents as taught by Shaw at the end of the assembly to enhance the lay out of the manifest in order to expedite information and facilitate efficiently the integrity checking as intended per module when resources at built time are available as mentioned in claim 1 above.

As per claim 6, Renaud in combination with Shaw, discloses version number included in the manifest (Fig. 3b) and security checking of modules and comparing of signatures identifier from the manifest and generated from the module (Fig. 4, 6); but does expressly mention determining if the contents of the assembly has been modified by comparing the actual hash from the module contents with the hash stored in the manifest. Notwithstanding the known concept that using hash and comparing hash for integrity checking as taught by both Renaud and Shaw, it is further noted via Shaw's comparison as to determine if the modules as hashed have been modified or tampered with the limitation as to comparing of an actual hash against hash of a listing manifest is disclosed. In case Renaud's hashed signature comparing does not already include a verification as to whether a module has been modified, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide a integrity checking such as taught by Shaw to the data authentication by Renaud's using the version number in the manifest as mentioned above to determine if the up/down-loaded module for activation are integral with respect to their predetermined version, thus enhancing further the integrity checking of data and security control as intended by both Renaud and Shaw.

As per claims 7 and 8, Renaud discloses version and publisher information (e.g. col. 6, lines 55-64) and thereby suggests which version determination but fails to expressly disclose if the assembly has been modified using such information (re claim 7) but this limitation has been addressed in claim 6 above. Further, Renaud does disclose whether the publisher of the assembly is (re claim 7) trustworthy (e.g. step 506 – Fig. 5, 5a; col. 9, lines 8-21) via authenticating publisher name information in the manifest(re claim 8).

As per claim 18, Renaud discloses a computer readable medium (col. 15, lines 13-30) with an executable for a runtime application program, such medium comprising

an assembly (e.g. *data structure 300* – Fig. 3a) including manifest containing a list of modules making up the assembly (e.g. Fig. 3A); and

a hash value of at least one modules in the assembly (e.g. identifier *316, 318* - Fig. 3B; col. 7, lines 15-27).

But Renaud does not specify that a hash value of one of module is hash of the contents of at least one module listed in the assembly and manifest. But this limitation has been addressed above in claim 1 using Shaw's teachings.

As per claim 23, this is a system claim version of claim 1 above; hence is rejected herein using the corresponding rejection set forth therein.

As per claim 24, this claim recites comparing hash of a module in the manifest and hash of actual module is analogous to the comparison limitation as recited in claim 6 above. Hence, the rejection in claim 6 herein applies.

As per claim 25, Renaud discloses identity, publisher and version information and Shaw discloses integrity checking of code assemblies or modules as mentioned in claims 3-4 and 6-8 when addressing trustworthiness of hash value, publisher and version information above. Based on such teachings and rationale of rejection as set forth therein, the limitation of using identity and version information to determine if the assembly should be executed would also have been obvious using the rationale of mainly claims 6-8.

4. Claims 10-17, 22, 27, and 29-35 are rejected under 35 U.S.C. 103(a) as being unpatentable over Renaud et al., USPN: 5,958,051, in view of Evans et al., USPN: 5,805,899 (hereinafter Evans); in view of Graunke et al., USPN: 5,991,399 (hereinafter Graunke).

As per claim 10, Renaud discloses a method for facilitating integrity checking employable by application programs at runtime (e.g. Fig. 6; col. 9, line 40 to col. 10, line 37), such method comprising:

providing an assembly (e.g. *data structure 300* – Fig. 3a) with manifest (Fig. 3b) that contains a list of assemblies (e.g. *data files 304-314, version of the file* - col. 6, lines 46-64; col. 7, lines 31-36 – Note: data files, digital stream or class files are assemblies of other sub-assemblies like subclasses or subdata) that make up the assembly;

providing a manifest (*signature file 302* – Fig. 3b)with a hash of the contents of at least one assemblies (e.g. *identifier, one-way hash* - col. 7, lines 15-27).

But Renaud does not disclose that the manifest contains a list of referenced assemblies that the assembly depends on; nor does Renaud disclose a manifest with a hash of a manifest of one referenced assembly of the list of referenced assemblies. However, Renaud teaches representing a manifest with a hashed representation (*signature 322* - Fig. 3B) and suggests dependency on multiple location (hashed) identifiers by one assembly at verification and runtime (e.g. col. 9, lines 30-39; *Add site* - Fig. 5a; col. 4, lines 9-34; Fig. 18). In a method to bind objects at runtime using hash representation of versioned assemblies analogous to the Renaud's signature file listing, Evans discloses pointer means to refer to other manifest of other assemblies that the pointing assembly depends on (re claim 9). In view of the known concept that manifest is a signed information structure submitted by manufacturer for identifying references to other hashed assemblies being delivered for integrity checking, such is taught by Graunke (e.g. col. 6, line 46 to col. 7, line 7), it would have been obvious for Renaud to combine the linked references as taught by Evans to provide manifest referencing other assemblies, such that each reference in turn contain manifest information, such information being hashed to enable integrity checking as taught by Graunke, because such manifest-like referencing information by Evans to point to other assemblies enable the integrity checking (manifest hashing or signed as by Graunke) of all referenced assemblies such as desired by Renaud for providing the correct set of trusted (or files or objects) and because of the integrity checking on manufacturer and other attributes as set forth by Graunke (col. 6, line 46 to col. 7, line 15)

Nor does Renaud teach analyzing the hash provided to the manifest and a second hash of the manifest of a referenced assembly at runtime to determine changes between runtime assembly and build time assembly. Based on the version information being part of the hashed data presented in a manifest as taught by Renaud, and the desirability to check version and manufacturer trustiness before using the downloaded code by Graunke, and further using Renaud's teaching to provide hash information in a manifest in order to check whether code can be trusted for updating or download, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide the dependencies linkage as taught by Evans to Renaud's hash listing and source location dependency to enhance Renaud/Graunke's multiple location/attributes dependency because hash of manifest content re-enforcing assemblies dependency checking prior to usage would enable a improve integrity checking and dependency of source data to retrieve according to the intention as contemplated by both Renaud and Graunke.

As for claim 11, the limitation would also have been obvious in view of the rationale as set forth in claim 10; because providing in a manifest a hash of each of the referenced assemblies would enable more time-efficient locating of depended-upon assemblies as well as verifying their integrity and authentication as intended by Renaud.

As per claims 12 and 13, Renaud also teaches identity information in the manifest (re claim 3) and publisher and version information (re claim 4).

As per claim 14, Renaud discloses hash of the identifiers of versioned modules in the assembly and suggests placing a signature (or hashed representation) of the manifest itself on top of the manifest with all other module identifiers at the end of the manifest (e.g. Fig. 3B), but **fails**

to disclose providing a hash of the contents of such assembly at the end of the assembly. But this limitation to provide hash of the content or signature at the bottom of listed components provided by the manufacturer (i.e. signed manifest) has been disclosed by Graunke (see *signature:<pk...>* - Fig. 3). Hence it would have been obvious for one of ordinary skill in the art at the time the invention was made to add such hash of referenced contents as taught by Graunke at the end of the assembly to enhance the lay out of the manifest in order to expedite information and facilitate efficiently the integrity checking as intended per module when resources at built time are available as mentioned in claim 1 above.

As per claim 15, the limitation as to determine whether the contents of the referenced assembly has been modified would also have been obvious in light of the rationale set forth in claim 6 which is further applied to the combination of Renaud/Evans/Graunke of claim 10 because this would enable enhancing further the integrity checking of data and security control as intended by both Renaud and Graunke.

As per claims 16 and 17, the rationales based on Graunke's teachings and used in the rejection of claim 10 are herein respectively applied to the combination of Renaud/Evans for the same motivation as set forth therein correspondingly.

As per claim 22, Renaud discloses a readable medium (re claim 18) such medium comprising an assembly with a manifest containing a list of referenced assemblies. But such limitations have been addressed in claim 10 and 14 above; using the corresponding rationale as set forth therein to address the referenced assemblies limitation and hash of contents of referenced assemblies(Note: Graunke integrity checking based on attributes of manufacturer reads on build time of assembly versus actual assembly hash value compared at runtime).

As per claim 27, Renaud discloses a method for facilitating integrity of assemblies employable by application program at runtime, such method comprising components for:

a first component to provide a manifest of assembly (e.g. *signature file 302* – Fig. 3b; *data structure 300* – Fig. 3a - Note : data structure 300 is assembly and signature file is manifest) that contains at least one a sub-assemblies (e.g. *data files 304-314, version of the file* - col. 6, lines 46-64; col. 7, lines 31-36 – Note: files, digital stream or class files are sub-assemblies making up assembly structure 300) that make up the assembly, such sub-assemblies comprise a manifest (e.g. Fig. a-b – Note: file representation by identifier 316, 318 is equivalent to manifest of each sub-assemblies); and

a second component to provide the manifest with a hash of the sub-assemblies (e.g. *identifier, one-way hash* - col. 7, lines 15-27 – Note: hash representation of file identifier is hash of the manifest of the sub-assemblies).

But Renaud does not discloses that the sub-assembly in the manifest is a referenced assembly; but this limitation has been addressed using the teachings by Evans to link assemblies to other assemblies reference using the rationale as set forth in claim 10 above.

Nor does Renaud disclose that such referenced assembly comprises a manifest. But this limitation would have been obvious as from above in view of Evans' teaching of a linked representation of referenced assemblies by a structure with manifest depiction of assembly identification information, i.e. each referenced assembly in turn including manifest information as to represent itself (e.g. structure # - Fig. 11).

Nor does Renaud disclose that the assembly manifest includes a hash of the manifest of at least one of the referenced assembly. The motivation to provide the assembly manifest with referenced assemblies having each a signed manifest (or hash of a manifest) and to provide the assembly manifest with a hashed information referencing other referenced assemblies would have been obvious for the same reasons as set forth in claim 10 above using Graunke.

Nor does Renaud teach a third component that compares hash of one referenced assembly with actual hash value of one referenced assembly to identify version changes. But this limitation would have been obvious in view of the combination Renaud/Graunke in light of Evans referencing of signed/hashed information of other referenced assemblies as set forth in claim 10 above.

As per claim 29, Renaud does not teach a binding policy but discloses establishing and verification of signature with version information included and access permissions protocol (e.g. Fig. 5, 5a, 7) to enable trusted communication to bind usage of received files to the secure and trusted protocols; and also teaches determining which applet is to be trusted for execution (Fig. 6). Evans discloses binding assemblies to another version representation of another assembly (Evans: Fig. 11) and Graunke disclose client/server trusted method to provide tamper resistant communication-based verification process for binding downloaded content with originating source attributes (see Fig. 2-5). It would have been obvious for one of ordinary skill in the art at the time the invention was made to modify Renaud's method for integrity checking using trusted protocol of versioned assemblies such as to enhance it with the version dependency linking and correct version determining by Evans or the client/server process by Graunke and effect it as a binding policy enabling a consistent process as to determining which version is to be executed when another version is resident on the executing environment; and which source provider can be trusted based on policy rules because this would enable alleviating system errors from having uncontrolled versions dynamically competing in a same system such as intended by Evans or the client/server software delivery tamper hazards as by Graunke (see text related to Evans, Fig. 11; and Graunke, Figs. 2-5).

As per claim 30, this claim is a means claim including limitations corresponding to referenced assembly limitations as recited in claim 27 above, and include integrity evaluating means for comparing hash as mentioned in claim 27, hence this claim is rejected with the same rejection as set forth in claim 27 above.

As per claim 31, Renaud teach about module (re claim 1) and Evans teaches about related assembly. The motivation to make such manifest of related assembly to represent the module of Renaud would have been obvious in light of the rationale in claim 27 above.

As per claims 32-33, refer to claim 27 for respective limitations.

As per claims 34, 35, refer to claims 29 and 21 respectively.

5. Claims 9, and 19-21 are rejected under 35 U.S.C. 103(a) as being unpatentable over Renaud et al., USPN: 5,958,051, and Shaw, USPN: 2002/0026634, as applied to claims 18, 23, and further in view of Evans et al., USPN: 5,805,899 and Graunke et al., USPN: 5,991,399.

As per claim 9, in reference to claim 1, Renaud discloses a signature or hashed representation of the manifest file (Fig. 3B) and identification of multiple sites signature (hashed representation) for a given loaded files (col. 9, lines 30-39; *Add site* - Fig. 5a) to verify the authenticity of the module in the loaded assembly (e.g. col. 4, lines 9-34) of another versioned object (Fig. 18) **but does not disclose** a manifest providing hash of another manifest upon which it depends on. Evans, in a method to link runtime versioned objects similar to the combination of components of Shaw's template using hash value of object identifier (Evans: Fig.

11) analogous to that such as taught by Renaud (Renaud: Fig. 4), discloses pointer information to refer to the assembly manifest of another versioned object that the assembly depends on (e.g. Fig. 16). A signed information structure submitted by manufacturer for identifying references to other hashed assemblies being delivered for integrity checking, and this is equivalent to a hash of a manifest, such is taught by Graunke (e.g. col. 6, line 46 to col. 7, line 7). In view of the source information being a manifest being hashed (or signed data) teachings by Graunke and hash dependency by Evans, combined with the hash representation of the manifest by Renaud, it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide the manifest file or signature file of Renaud, with pointer information to locate the site of another manifest site or assembly that the pointing assembly depends on, such location or reference listing represented by a hashed identifier as suggested by Graunke. One of ordinary skill in the art would be motivated to do so because this would extend the security/version checking adopted by Renaud in that it reduces time to locate a referred to site of another assembly of modules, such reduction of time being enhanced by using source location information as taught by Shaw, to locate and verify the hashed representation of the depended-upon assembly (hash of the manifest) as suggested by Graunke enabling improved integrity checking of referenced assemblies based on manufacturer attributes pointed to by the dependency scheme from Evans.

As per claim 19, this claim includes a manifest with a list of at least one referenced assembly.

Renaud teaches representing a manifest with a hashed representation (*signature 322* - Fig. 3B) and suggests dependency on multiple location (hashed) identifiers by one assembly at verification and runtime (e.g. col. 9, lines 30-39; *Add site* - Fig. 5a; col. 4, lines 9-34; Fig. 18). In a method to bind objects at runtime using hash representation of versioned assemblies analogous to the Renaud's signature file listing, Evans discloses pointer means to refer to other manifest of other assemblies that the pointing assembly depends on (re claim 9). Hence, it would have been obvious for one of ordinary skill in the art at the time the invention was made to modify the representation of manifest by Renaud with the implementation suggested by Evans to point to other assemblies referenced to by the pointing assembly in providing a manifest with the list of referenced assemblies (or files or objects) and using therein pointer information to point to the hash representation of the manifest of the assembly referred to by such pointer because of the same benefits as set forth in claim 9 above.

Renaud does not disclose a hash of manifest of such referenced assembly. But this limitation has been addressed using the teaching of Graunke as set forth in claim 10; and combined with the reference assembly teaching by Evans, would have been obvious according to the rationale as set forth in claim 10.

As per claim 20, refer to rejection of claim 4.

As per claim 21, Renaud mentions about application classes used in a GUI window environment (e.g. Fig. 5a) to provide interactive verification of applets and Shaw discloses template storage DLL to enhance the component builder (col. 7, lines 48-61). Official notice is taken that the use of Dynamic Linked Library to effect windows application interfaces or/and browser/user interface functionality; or to support system low-level calls was a well-known concept at the time of the invention. In case the components to build by Shaw or the assembly of code or modules by Renaud do not already include a Dynamic Linked Library (DLL), it would have been obvious for one of ordinary skill in the art at the time the invention was made to provide a DLL mentioned in the above notice as one type of assembly of code in the list of assemblies as taught by Renaud/Shaw, because of the known benefits ascribed to using DLL such as portability, storage benefits and ease to use without the need for recompilation.

6. Claim 26 is rejected under 35 U.S.C. 103(a) as being unpatentable over Renaud et al., USPN: 5,958,051, and Shaw, USPN: 2002/0026634, as applied to claim 23, and further in view of Graunke et al., USPN: 5,991,399.

As per claim 26, with reference to claim 23, Renaud does not expressly teach a binding policy but discloses establishing and verification of signature with version information included and access permissions protocol (e.g. Fig. 5, 5a, 7) to enable trusted communication to bind usage of received files to the secure and trusted protocols; and also teaches determining which applet is to be trusted for execution (Fig. 6). In building components using secure object verification means, Shaw further enhances Renaud's version recording by disclosing integrity checking to verify if data are not corrupted or modified (*check license and integrity, component has been damaged* - col. 17, lines 16-44). Graunke disclose client/server trusted method to provide tamper resistant communication-based verification process for binding downloaded content with originating source attributes, among which versions of manufactured content (see Fig. 2-5). It would have been obvious for one of ordinary skill in the art at the time the invention was made to modify Renaud's (combined with Shaw) method for integrity checking using trusted protocol of versioned assemblies such as to enhance it with the version binding and linking at runtime from the client/server process by Graunke and effect it as a binding policy enabling a consistent process as to determining which version is to be executed when another version is resident on the executing environment; and which source provider can be trusted based on policy rules because this would enable alleviating system errors from having uncontrolled versions dynamically competing in a same system such as intended in the client/server software delivery tamper hazards as by Graunke (see text related to Graunke, Figs. 2-5) or as raised as concerns in the Renaud or Shaw's BACKGROUND.

Response to Arguments

7. Applicant's arguments with respect to claims 1-35 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

8. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (272) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Kakali Chaki can be reached on (571)272-3719.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence – please consult Examiner before using) or 703-872-9306 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

VAT
April 16, 2005

[illegible]

KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100